Dependency Injection

حد فيكم واجه مشكلة قبل كده لما كان بيحاول يعدّل في كود معقد جدا وبالاخص مثلا فى ليله الديدلاين ؟ (:

تخيل معايا إنك بتبني بيت وبدل ما تستعين بسباك تقرر تبني مواسير الميه بنفسك، وتلحمها، وتصلح كل مشكلة فيها بنفسك!
بعدين تيجي توصل الكهربا فتروح تشتري الأسلاك، وتركبها بنفسك، ويمكن كمان تصنع الفيش والمفاتيح من الصفر

مجهود رهيب، صح؟
طب ما هو ده بالضبط اللي بيحصل في الكود لما منستخدمش *Dependency Injection*
بدلاً من إن الكود يكون مرن وقابل لإعادة الاستخدام، بنلاقي نفسنا بنبني كل حاجة من الصفر في كل مرة

طيب يعني إيه Dependency Injection أصلاً؟

ببساطة، الـ Dependency Injection أو DI هو طريقة منظمة لإدارة الـ Dependencies في الكود بتاعنا. لما يبقى عندك كلاس بيعتمد على كلاس تاني، بدل ما تروح تنشئ الكلاس التاني جوه الأوكلاني بنفسك، الـ DI بيسمحلك تمرر الـ dependency من الخارج، بحيث يبقى الكود أكتر مرونة وأسهل في التعديل والاختبار."

بمعنى تاني، بدل ما الكلاس يقول: أنا هروح أجيب الحاجة اللي محتاجها الكلاس ببساطة بيقول هاتولي الحاجة اللي محتاجها وأنا هشتغل بيها
وده بيخلي الكود بتاعك Modular، نظيف، وسهل التطوير والصيانة.

طيب ليه ده مهم؟

"تخيل معايا إنك شغال في شركة كبيرة، والكود بتاعك لازم يكون قابل للتطوير على المدى البعيد. بدون DI، أي تعديل في كلاس واحد ممكن يبوظ الدنيا كلها! لكن مع DI، تقدر تبدّل أي جزء في السيستم بسهولة بدون ما تخاف إنه يكسر حاجة تانية.

كمان لما تيجي تختبر الكود لو كل حاجة مربوطة ببعضها بشكل مباشر، هتلاقي نفسك محتاج تشغل الكود كله عشان تختبر جزء صغير منه، وده كارثة لكن مع DI، تقدر تستخدم Mocks أو Stubs وتختبر أي جزء من الكود لوحده من غير مشاكل.

إيه اللي هيحصل لو مستخدمناش Dependency Injection؟ الكارثة بتبدأ هنا!

دلوقتي، تخيل إنك بتكتب كود لمشروع كبير عندك كلاس اسمه Car، وطبعا العربية محتاجة Engine علشان تشتغل، فإيه اللي ممكن يحصل لو كتبنا الكود بالطريقة التقليدية؟"_

١- تداخل الكود بشكل مبالغ فيه (Tightly Coupled Code)

بدون Dependency Injection، كلاس `Car` هيكون مسؤول عن إنشاء `Engine` بنفسه، بالشكل ده:

```
#include <iostream>
using namespace std;


class Engine {
public:
    void start() {
        cout << "Engine started!" << endl;
    }
};


class Car {
private:
    Engine engine;  // هنواجه مشكلة Engine تداخل قوي، لو حبينا نغير نوع الـ

public:
    void drive() {
        engine.start();
        cout << "Car is moving..." << endl;
    }
};

int main() {
    Car car;
    car.drive();
    return 0;
}
```

ليه الكود ده مشكلة؟

1. **غير مرن** – لو حبينا نغير `Engine` بـ `ElectricEngine` لازم نعدل كود `Car` نفسه.
2. **صعب في الاختبار** – `Car` بيعمل `Engine` داخله، فمش هينفع نستخدم Mock أو نختبره لوحده بسهولة.

طيب هنحل ده ازاى ؟؟؟؟؟؟؟؟

to be continued